REMARKS

A. Overview

Claims 1-59 were pending when the Office Action was mailed. The Office Action rejects all of the claims. Applicants herein amend claims 1-3, 6, 7, 18, 21, 22, 33, 46, 49 and 51-55 and cancel claims 14, 15, 17, 29, 30, 32, 34, 35, 37, 50, and 55-59. Accordingly, claims 1-13, 16, 18-28, 31, 33, 36, 38-49, and 51-54 remain pending.

Applicants would like to thank the Examiner for the consideration extended during the telephone interview conducted on February 20, 2008. During the interview, Examiner Zhen and the undersigned discussed applicants' technology, the Hagan reference, and proposed claim amendments. Should the Examiner need additional information regarding the telephone interview, he is asked to contact the undersigned.

The Office Action provisionally rejects claims 1-9 and 18-24 under 35 U.S.C. § 101; rejects claims 18, 21, 23, 25, and 29-32 under 35 U.S.C. § 102(e) over Hagan; and rejects claims 1-17, 19, 20, 22, 24, 26-28, and 33-59 under 35 U.S.C. § 103(a) over a combination of Hagan and Schimmel. Applicants respectfully traverse these rejections. Nevertheless, applicants herein amend the claims to clarify the claimed subject matter.

B. Rejection under 35 U.S.C. § 101

Claims 1-9 and 18-24 are provisionally rejected under 35 U.S.C. § 101 as claiming the same invention as claims 22-37 of copending Application No. 10/366,037 and claims 22-37 of copending Application No. 10/339,366. As discussed during the telephonic interview, applicants note that each of the above-mentioned applications have been abandoned. Accordingly, applicants respectfully request that the Examiner reconsider and withdraw these rejections.

C. Prior Art Rejections

Applicants' technology enables threads in a multithreaded environment to access a single collection of items in parallel. (Specification at Abstract, ¶ [0082]). The disclosed technology manages the collection using a write counter, a read counter, a lower bound, a bucket array, and a linked list associated with each bucket. (Specification at ¶ [0082], Figure 12). The read and write counters reference the buckets to which the next read or write will occur and the lower bound indicates a lower bound on the number of items in the collection. (Specification at ¶ [0082]). Threads add and remove items from the bucket array in a circular manner. (*Id.*). That is, the j^{th} item added to the collection is added to bucket[j modulo N] and the k^{th} item read from the collection is read from bucket[k modulo N], where N is the number of buckets in the array. (*Id.*). Access to each bucket is synchronized using a synchronization mode of sync and a full/empty bit associated with each bucket. When a bucket's full/empty bit is set to full and a thread attempts to read from the bucket using a synchronization access mode of sync, the thread is permitted to access the bucket and the bit is set to empty. (Specification, ¶¶ [0084]-[0085]). However, if a thread attempts a read of a bucket whose full/empty bit is set to empty, the thread is blocked from accessing the bucket until the bit is set to full. Synchronized write instructions operate in just the opposite manner. That is, when a thread attempts a synchronized write of a bucket whose full/empty bit is set to full, the thread is blocked from accessing the bucket. When a thread attempts a synchronized write of a bucket whose full/empty bit is set to empty, the thread is permitted to write to the bucket and the bit is set to full. In short, the full/empty indicates whether a bucket is available for consumption or available to be written to. The technology allows parallel access to the collection by multiple threads by effectively locking only a portion of the collection (i.e., represented by a bucket) at a time. Thus, each bucket can be locked and accessed by different threads at the same time using a synchronization access mode of sync. This allows the different threads to

access the different portions of the collection without waiting for other threads to finish their access.

Hagan is directed to techniques for handling a plurality of queues within a data processing system. (Hagan, Abstract; 2:2-8). In Hagan, each queue stores a different collection of tasks corresponding to a specific priority level; tasks of one queue are processed only after each task of a higher priority level queue are processed. (Hagan, 6:10-26). Hagan processes queues exclusively, "ensuring that only one queue is being processed while others remained locked." (Hagan, 8:61-64). Schimmel is directed to techniques for dynamically resizing a hash table when the number of records stored therein exceeds a maximum number. (Schimmel, Abstract; 4:1-8).

Claim 1 now recites "reading from the bucket pointed to by the fetched write counter using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to." Similarly, claim 18 now recites "reading from the bucket pointed to by the fetched read counter using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to." Similarly, claim 33 now recites "reading from the bucket at the location defined by the pointer using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to." Similarly, claim 49 recites "a component that accesses the bucket at the location defined by the pointer using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to."

Neither Hagan nor Schimmel disclose these features. The Office Action points to Hagan 8:57-9:14 as disclosing "reading from the bucket pointed to by the fetched write pointer using a synchronization access mode of sync." (Office Action, Page 6). Applicants' claimed technology locks a portion of a collection of items (i.e., the portion corresponding to a bucket) while a thread accesses that portion, but does not lock other portions of the collection, allowing other threads to access these portions of the collection of items. The relied upon portion of Hagan, however, describes locking an entire collection of items while one is being processed. Hagan fails to teach or suggest a technique for managing a collection of items in a multithreaded environment by locking only a portion (i.e., a bucket) of the collection, as recited. Furthermore, Hagan does not disclose a synchronization technique that allows one thread to gain exclusive read access to a resource by setting a bit associated with the resource to one value and exclusive write access to the resource by setting the bit to another value. Accordingly, claims 1, 18, 33, and 49 are patentable over the cited prior art, as are their dependent claims.

D. Conclusion

In view of the above amendments and remarks, applicants believe the pending application is in condition for allowance and request reconsideration.

Please charge any deficiency in fees or credit any overpayment to our Deposit Account No. 50-0665, under Order No. 324758003US6 from which the undersigned is authorized to draw.

Dated: February 27, 2008

Respectfully submitted,

By /Maurice J Pirio/

Maurice J. Pirio
Registration No.: 33,273
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorney for Applicant